

**Amendments to the Specification:**

**Replace the paragraph on page 4 lines 16-18 with:**

C1  
In one embodiment, a method comprises: receiving data having a plurality of N bytes: [D(0), D(1), ... ,D(N-1)] each byte having a parity bit  $\bar{p}_P$ ; and computing the parity of [P(0), P(1), ...P(N-1)].

**Replace the Paragraphs on Page 35, beginning at line 22 with:**

More particularly, the DATA, memory control, ADDR, and "tag" portions (with their byte parity ( $\bar{p}_P$ ) generated by parity generator 5102 (FIGS. 11A, 11B, 11C and 11D)) of the information coupled to the output of selector 5120 is stored in the register 5124. As noted above in connection with FIG. 16, the DATA\_CRC portion (i.e., the words X and Y) occurs after the last DATA word.

C2  
Thus, as the words in the DATA clock through register 5124 they pass into the DATA\_CRC checker 5132 where the CRC of the DATA is determined (i.e., the DATA\_CRC checker 5132 determine X and Y words of the DATA fed to such checker 5132). The actual X and Y words (i.e., DATA\_CRC stored in register 5128, both content (n) and parity (p)) are stored successively in register 5128 and are then passed to checker 5132 where they are checked against the X and Y words determined by the checker 5132. As noted above, the DATA has appended to it its parity ( $\bar{p}_P$ ). Thus, the "information" whether in register 5124 or register 5128 has a content portion indicated by "n" and its parity indicated by " $\bar{p}_P$ ". Thus, the DATA\_CRC register 5128 includes the DATA\_CRC previously stored in register 5104<sub>1</sub> (FIGS. 11A, 11B, 11C and 11D) (i.e., the content portion designated by "n") and its parity (designated by "p"). The DATA, memory control, ADDR, and "tag" portions, (with their parity ( $\bar{p}_P$ ) (i.e., content "n" plus its appended parity " $\bar{p}_P$ ") stored in register 5124 may be coupled through a selector 5149 through one of two paths: One path is a direct path when the "Wait and Validate" command is not issued by the director; and, a second path which includes a delay network

5130, here a three clock pulse delay network 5130.

More particularly, it is noted that the DATA, control, ADDR, "tag", both content (n) and parity ( $\bar{p}$  P) are also fed to a DATA\_CRC checker 5132. Also fed to the DATA\_CRC checker 5132 is the output of DATA\_CRC register 5128. The CRC checker 5132 checks whether the DATA\_CRC (content "n" plus its parity "p") is the same as the CRC of the DATA, such DATA having been previously stored in register 5104<sub>2</sub> (FIGS. 11A, 11B, 11C and 11D), i.e., the content "n" plus its parity " $\bar{p}$  P" of the DATA previously stored in register 5104<sub>2</sub> (FIGS. 11A, 11B, 11C and 11D). If they are the same, (i.e., no DATA\_CRC\_ERROR), a logic 0 is produced by the CRC checker 5132. If, on the other hand, they are not the same, (i.e., a DATA\_CRC\_ERROR), the CRC checker 5132 produces a logic 1. The output of the Data\_CRC checker 5132 thereby indicates whether there is an error in the CRC of the DATA. Note that a DATA\_CRC\_ERROR is not known until three clock cycles after the last sixteen-bit portion of the DATA (i.e., the word of the DATA, FIG. 16) is calculated due to the nature of the CRC algorithm. Such indication is fed to a selector 5152 via an OR gate 5141. If there is a DATA\_CRC\_ERROR, the "information" at the output of the delay network 5130 (i.e., the last word of the DATA (FIG. 16)) with its parity ( $\bar{p}$  P) is corrupted. Here, the content (n) of such "information" (i.e., the "information" at the output of the delay network 5130 (i.e., the last word of the DATA (FIG. 16))) is fed to a second input I<sub>2</sub> of the selector 5140. The parity ( $\bar{p}$  P) of such "information" (i.e., the last word of the DATA (FIG. 16)) is fed non-inverted to one input of selector 5152 and inverted, via inverter 5150, to a second input of the selector 5152. If there is a DATA\_CRC\_ERROR detected by data CRC checker 5132, the inverted parity is passed through the selector 5152 and appended to the content portion (n) of the "information" (i.e., the last word of the DATA (FIG. 16)) provided at the output of the delay network 5130 and both "n" and appended " $\bar{p}$  P" are fed to the second input I<sub>2</sub> of selector 5140 thereby corrupting such "information". It should be noted that the remaining portions of the information cycle (i.e., the memory control, address (ADDR), "tag", and all but the last word of the DATA (FIG. 16)) pass through the delay network 5130 without having their parity ( $\bar{p}$  P)

corrupted.

C2

If there is a no "Wait and Validate" transfer, logic decoder 5122 selects the first input  $I_1$  as the output of the selector 5140. If there is a "Wait and Validate" transfer, the logic decoder 5122 selects the second input  $I_2$  as the output of the selector 5140. It is noted, however, that because the last word of DATA (FIG. 16) is delayed three clock pulses (from Clock 1) by registers 5142, 5144, and 5146 (such registers 5142, 5144 and 5146 being fed by such Clock 1), the DATA\_CRC check is performed before the last word of the DATA appears at the output of register 5146. Thus, the last word of the DATA is corrupted in byte parity before being passed to the logic section 5010<sub>1</sub>-5010<sub>8</sub>. That is, because of the delay network 5130, the DATA\_CRC is evaluated before the last word of the DATA has passed to port 5008<sub>1</sub>. This corruption in parity (p), as a result of a detected DATA\_CRC error, is detected by a parity checker 6106 (FIGS. 14A, 14B, 14C and 14D) in the following logic section 5010<sub>1</sub>-5010<sub>8</sub> in a manner to be described. Suffice it to say here, however, that detection of the parity error (produced by the detected CRC error) prevents such corrupted information from storage in the SDRAMs.

On the other hand, if there is no DATA\_CRC\_ERROR (and no error in the parity of the DATA\_CRC detected by the parity checker 6106 (FIGS. 14A, 14B, 14C and 14D) in a manner to be described) the non-inverted parity (p) is appended to the "information" (i.e., DATA, memory control, ADDR, and "tag") provided at the output of the delay network 5130 and such information is fed to the proper memory address region  $R_1$ - $R_4$  as indicated by "tag".

---

**Replace the Paragraphs on Page 38, beginning at line 28 with:**

---

C3

Thus, the exemplary lower port interface section W (FIGS. 12A, 12B, 12C and 12D) includes a parity generator made up of an exclusive OR gate 5134 and register 5136 arranged as shown fed by the parity ( $\bar{p}_P$ ) of the DATA portion stored in register 5124. The generated parity  $\bar{p}_P$  is fed to a comparator 5138 along with the parity (p) of the DATA\_CRC (i.e., DATA\_CRC\_PARITY), as indicated. If the two are the same at the end of the DATA portion of the information cycle (FIG. 16), a logic 0 is produced by the

comparator 5138 and such logic 0 passes to the selector 5152 to enable the non-inverted parity to pass through such selector 5152. If there is an error in the parity bit of the CRC, a logic 1 is produced by the comparator 5138 and the inverted parity is passed through the selector 5152. The logic 1 output of comparator 5138 passes through OR gate 5141 to couple the inverted parity (p) through selector 5152 to append to the content port (n) of DATA control, ADDR, and "tag" at port I<sub>2</sub> of selector 5140. Thus, if there is either a DATA\_CRC\_ERROR or if DATA\_CRC\_PARITY is different from parity of the DATA\_PARITY at the end of the DATA portion of the information cycle as indicated by a signal produced on line COMP\_ENABLE by the logic decoder 5122, a logic 1 is produced at the output of OR gate 5141 thereby coupling the inverted parity through selector 5152. Otherwise, the non-inverted parity passes through selector 5152. That is, the COMP\_EN is produced at the end of the DATA in the information cycle (FIG. 16).

C3  
It is noted that information read from the memory region passes to a register 5170 and a CRC generator 5172. The generated CRC is appended to the information clocked out of the register 5170. Four copies of the information with appended CRC are stored in registers 5174<sub>1</sub>-5174<sub>4</sub>, respectively. In response to the "tag" portion fed to logic decoder 5122, a selected one of the registers 5174<sub>1</sub>-5174<sub>4</sub> is coupled to one of the port W<sub>1</sub>-W<sub>4</sub> by selector 5180 and gates 5182<sub>1</sub>-5182<sub>4</sub> in a manner similar to that described in connection with FIGS. 11A, 11B, 11C and 11D.

Referring now to FIGS. 13A, 13B, 13C, 13D and 13E, a pair of the logic sections 5010<sub>1</sub>-5010<sub>8</sub> (memory array region controllers), here logic sections 5010<sub>1</sub> and 5010<sub>2</sub> are shown. As noted above in connection with FIGS. 9A, 9B and 9C, both logic sections 5010<sub>1</sub> and 5010<sub>2</sub> are coupled to the same memory array region, here memory array region R<sub>1</sub>. As was also noted above in connection with FIGS. 9A, 9B and 9C, the logic section 5010<sub>1</sub> is in one fault domain, here fault domain A, and logic section 5010<sub>2</sub> is in a different fault domain, here fault domain B. Thus, logic section 5010<sub>1</sub> operates in response to clock pulses from Clock 1 and logic section 5010<sub>2</sub> operates in response to clock pulses from Clock 2.

As noted above, each logic section 5010<sub>1</sub>-5010<sub>8</sub> (FIGS. 9A, 9B and 9C) includes a

C3 pair of upper ports, A and B, a control port C and a data port D. Referring to FIGS. 13A, 13B, 13C, 13D and 13E, an exemplary logic section 5010<sub>1</sub> is shown in detail to include a upper port A controller 6002A coupled to upper port A, a upper port B controller 6002B coupled to upper port B, and a memory refresh section 6002R.

---

**Replace the paragraphs beginning on page 38 lines 14-25 with:**

---

C4 It has been discovered that the parity (p) of the DATA\_CRC must be the same as the parity of the DATA parity ( $\bar{p}P$ ). Thus, one merely has to check whether the parity of the DATA\_CRC is the same as the parity of the DATA parity ( $\bar{p}P$ ). Therefore, such detection DATA\_CRC parity checking method is accomplished without using the DATA\_CRC itself.

More particularly, since the DATA over which the DATA\_CRC is being calculated is already parity protected, one can use the DATA parity ( $\bar{p}P$ ) to calculate the DATA\_CRC parity: i.e., the DATA\_CRC parity is equal to the parity of all the DATA parity bits. Still more particularly, if there are N bytes of DATA:

$[D(0), D(1), \dots D(N-1)]$

and each byte is protected by a parity bit  $\bar{p}P$ , then the DATA\_CRC parity is the parity of

$[\bar{p}P(0), \bar{p}P(1), \dots \bar{p}P(N-1)]$ .

---